(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 114640681 B
(45)授权公告日 2024.05.17

(21)申请号 202210235673.X

(22)申请日 2022.03.10

(65)同一申请的已公布的文献号
申请公布号 CN 114640681 A

(43)申请公布日 2022.06.17

(73)专利权人 京东科技信息技术有限公司
地址 100176 北京市北京经济技术开发区科创十一街18号院2号楼6层601

(72)发明人 赵天云 陈蒙

(74)专利代理机构 中原信达知识产权代理有限责任公司 11219
专利代理师 郝红玉 张春晓

(51)Int.Cl.
H04L 67/1008(2022.01)
H04L 67/1012(2022.01)
H04L 67/1029(2022.01)
H04L 67/1031(2022.01)
H04L 41/0631(2022.01)
H04L 43/0829(2022.01)
H04L 43/16(2022.01)

(56)对比文件
CN 102521050 A,2012.06.27
CN 111538586 A,2020.08.14
CN 112363827 A,2021.02.12
CN 112788076 A,2021.05.11
CN 112988390 A,2021.06.18
CN 113691611 A,2021.11.23
CN 113849312 A,2021.12.28
CN 113918344 A,2022.01.11
US 2014068611 A1,2014.03.06
WO 2018001004 A1,2018.01.04
CN 105975334 A,2016.09.28
CN 107688495 A,2018.02.13
CN 107977271 A,2018.05.01
韩仲海;张曦煌.Web集群中基于负载均衡的QoS-aware请求调度算法.计算机工程与设计.2007,(第07期),全文.
周顺.面向Web Service的动态负载均衡设计与实现.计算机工程与科学.2010,(第12期),全文.

审查员 李玥

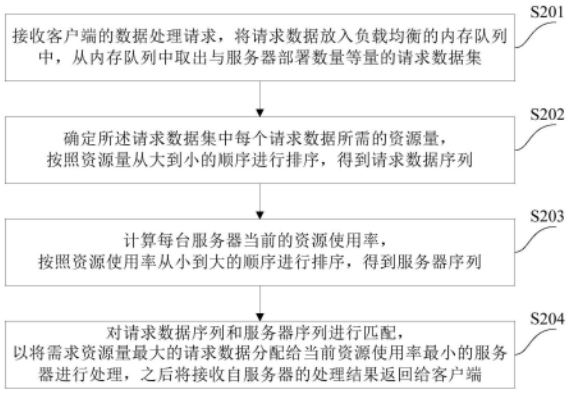
权利要求书2页 说明书8页 附图4页

(54)发明名称

一种数据处理方法和系统

(57)摘要

本发明公开了一种数据处理方法和系统,涉及计算机技术领域。该方法的一具体实施方式包括:接收客户端的数据处理请求,将请求数据放入负载均衡的内存队列中,从内存队列中取出与服务器部署数量等量的请求数据集;确定请求数据集中每个请求数据所需的资源量,按照资源量从大到小的顺序进行排序,得到请求数据序列;计算每台服务器当前的资源使用率,按照资源使用率从小到大的顺序进行排序,得到服务器序列;对请求数据序列和服务器序列进行匹配,以将需求资源量最大的请求数据分配给当前资源使用率最小的服务器进行处理,将接收自服务器的处理结果返回给客户端。该实施方式采用负载均衡方式调度TF-Serving,达到合理使用GPU资源、提高TF-Serving高可用性的目的。



CN 114640681 B

1.一种数据处理方法,其特征在于,包括:

通过客户端服务模块接收客户端的数据处理请求,将请求数据放入负载均衡的内存队列中,从内存队列中取出与服务器部署数量等量的请求数据集;其中,将客户端服务模块和服务器部署到同一pod中的不同docker容器内;

确定所述请求数据集中每个请求数据所需的资源量,按照资源量从大到小的顺序进行排序,得到请求数据序列;

计算每台服务器当前的资源使用率,按照资源使用率从小到大的顺序进行排序,得到服务器序列;其中,使用如下公式计算每台服务器的资源使用率:CPU使用率*第一权重+内存使用率*第二权重+GPU使用率*第三权重=服务器资源使用率,且第一权重、第二权重、第三权重的和为1;

对请求数据序列和服务器序列进行匹配,以将需求资源量最大的请求数据分配给当前资源使用率最小的服务器进行处理,之后将接收自服务器的处理结果返回给客户端。

2.根据权利要求1所述的方法,其特征在于,所述计算每台服务器当前的资源使用率,包括:

调用监控服务器程序的接口,以获取每个服务器当前的资源使用情况,进而计算每台服务器的资源使用率;其中,资源使用情况至少包括GPU、CPU和内存的使用情况。

3.根据权利要求1所述的方法,其特征在于,还包括:

若所述当前资源使用率最小的服务器的可用资源量,小于最大资源需求量,则触发服务器执行丢包操作,以将需求资源量最大的请求数据丢弃。

4.根据权利要求3所述的方法,其特征在于,还包括:

若任一服务器在预设时间段内丢包的数量大于或等于预设数值,则触发报警提醒,以对所述任一服务器添加资源量、和/或对请求数据进行限流操作。

5.根据权利要求1-4中任一项所述的方法,其特征在于,与客户端之间采用TCP长连接方式进行交互通讯。

6.一种数据处理系统,其特征在于,包括客户端服务模块、负载均衡服务模块和服务器部署模块,将客户端服务模块和服务器部署到同一pod中的不同docker容器内;

所述客户端服务模块,用于提供微服务接口供客户端调用,接收客户端的数据处理请求,将请求数据放入负载均衡服务模块的内存队列中;以及将所述负载均衡服务模块传输的处理结果返回给客户端;

所述负载均衡服务模块,用于从内存队列中取出与服务器部署数量等量的请求数据集,确定所述请求数据集中每个请求数据所需的资源量,按照资源量从大到小的顺序进行排序,得到请求数据序列;计算每台服务器当前的资源使用率,按照资源使用率从小到大的顺序进行排序,得到服务器序列;对请求数据序列和服务器序列进行匹配,以将需求资源量最大的请求数据分配给当前资源使用率最小的服务器;其中,使用如下公式计算每台服务器的资源使用率:CPU使用率*第一权重+内存使用率*第二权重+GPU使用率*第三权重=服务器资源使用率,且第一权重、第二权重、第三权重的和为1;

所述服务器部署模块,用于处理所述负载均衡服务模块传输的请求数据,之后将处理结果通过所述负载均衡服务模块发给所述客户端服务模块。

7.根据权利要求6所述的系统,其特征在于,还包括服务器监控模块和服务器使用率模

块;

所述服务器监控模块,用于获取每台服务器的资源使用情况;其中,资源使用情况至少包括GPU、CPU和内存的使用情况;

所述服务器使用率模块,用于从所述服务器监控模块中获取资源使用情况,结合每种资源的预设权重,计算每台服务器的资源使用率;

所述负载均衡服务模块,用于将当前资源使用率最小的服务器,作为处理所需资源量最大的请求数据的目标服务器。

8.根据权利要求6或7所述的系统,其特征在于,所述客户端服务模块中的接口参数设置为集合参数,以批量接收请求数据。

9.一种电子设备,其特征在于,包括:

一个或多个处理器;

存储装置,用于存储一个或多个程序,

当所述一个或多个程序被所述一个或多个处理器执行,使得所述一个或多个处理器实现如权利要求1-5中任一所述的方法。

10.一种计算机可读介质,其上存储有计算机程序,其特征在于,所述程序被处理器执行时实现如权利要求1-5中任一所述的方法。

一种数据处理方法和系统

技术领域

[0001] 本发明涉及计算机技术领域,尤其涉及一种数据处理方法和系统。

背景技术

[0002] 深度学习的底层大多采用C++开发,但为方便模型在工程过程中的使用,深度学习框架为不同的开发语言提供了对应的工具包。目前主要采用两种方式:1、通过深度学习框架提供的工具包,直接调用持久化的模型文件;2、通过TF-Serving或其他在线服务部署模型。

[0003] 但这两种部署方式具有缺点,如:受语言及开发人员水平的限制,使用GPU批处理时较为复杂,服务可用度较低;无法充分利用GPU算力的优势导致部署费用昂贵,以及不能合理使用GPU资源,也无法集群化部署服务等。

发明内容

[0004] 有鉴于此,本发明实施例提供一种数据处理方法和系统,至少能够解决现有技术中GPU微服务并发性能较低的现象。

[0005] 为实现上述目的,根据本发明实施例的一个方面,提供了一种数据处理方法,包括:

[0006] 接收客户端的数据处理请求,将请求数据放入负载均衡的内存队列中,从内存队列中取出与服务器部署数量等量的请求数据集;

[0007] 确定所述请求数据集中每个请求数据所需的资源量,按照资源量从大到小的顺序进行排序,得到请求数据序列;

[0008] 计算每台服务器当前的资源使用率,按照资源使用率从小到大的顺序进行排序,得到服务器序列;

[0009] 对请求数据序列和服务器序列进行匹配,以将需求资源量最大的请求数据分配给当前资源使用率最小的服务器进行处理,之后将接收自服务器的处理结果返回给客户端。

[0010] 可选的,所述计算每台服务器当前的资源使用率,包括:

[0011] 调用监控服务器程序的接口,以获取每个服务器当前的资源使用情况,进而计算每台服务器的资源使用率;其中,资源使用情况至少包括GPU、CPU和内存的使用情况。

[0012] 可选的,还包括:若所述当前资源使用率最小的服务器的可用资源量,小于最大资源需求量,则触发服务器执行丢包操作,以将需求资源量最大的请求数据丢弃。

[0013] 可选的,还包括:若任一服务器在预设时间段内丢包的数量大于或等于预设数值,则触发报警提醒,以对所述任一服务器添加资源量、和/或对请求数据进行限流操作。

[0014] 可选的,与客户端之间采用TCP长连接方式进行交互通讯。

[0015] 为实现上述目的,根据本发明实施例的另一方面,提供了一种数据处理系统,包括客户端服务模块、负载均衡服务模块和服务器部署模块;

[0016] 所述客户端服务模块,用于提供微服务接口供客户端调用,接收客户端的数据处

理请求,将请求数据放入负载均衡服务模块的内存队列中;以及将所述负载均衡服务模块传输的处理结果返回给客户端;

[0017] 所述负载均衡服务模块,用于从内存队列中取出与服务器部署数量等量的请求数据集,确定所述请求数据集中每个请求数据所需的资源量,按照资源量从大到小的顺序进行排序,得到请求数据序列;计算每台服务器当前的资源使用率,按照资源使用率从小到大的顺序进行排序,得到服务器序列;对请求数据序列和服务器序列进行匹配,以将需求资源量最大的请求数据分配给当前资源使用率最小的服务器;

[0018] 所述服务器部署模块,用于处理所述负载均衡服务模块传输的请求数据,之后将处理结果通过所述负载均衡服务模块发给所述客户端服务模块。

[0019] 可选的,还包括服务器监控模块和服务器使用率模块;

[0020] 所述服务器监控模块,用于获取每台服务器的资源使用情况;其中,资源使用情况至少包括GPU、CPU和内存的使用情况;

[0021] 所述服务器使用率模块,用于从所述服务器监控模块中获取资源使用情况,结合每种资源的预设权重,计算每台服务器的资源使用率;

[0022] 所述负载均衡服务模块,用于将当前资源使用率最小的服务器,作为处理所需资源量最大的请求数据的目标服务器。

[0023] 可选的,所述服务器监控模块,还用于:若所述当前资源使用率最小的服务器的可用资源量,小于最大资源需求量,则触发服务器执行丢包操作,以将需求资源量最大的请求数据丢弃。

[0024] 可选的,所述服务器监控模块,还用于:若任一服务器在预设时间段内丢包的数量大于或等于预设数值,则触发报警提醒,以对所述任一服务器添加资源量、和/或对请求数据进行限流操作。

[0025] 可选的,所述客户端服务模块中的接口参数设置为集合参数,以批量接收请求数据。

[0026] 可选的,与客户端之间采用TCP长连接方式进行交互通讯。

[0027] 为实现上述目的,根据本发明实施例的再一方面,提供了一种数据处理电子设备。

[0028] 本发明实施例的电子设备包括:一个或多个处理器;存储装置,用于存储一个或多个程序,当所述一个或多个程序被所述一个或多个处理器执行,使得所述一个或多个处理器实现上述任一所述的数据处理方法。

[0029] 为实现上述目的,根据本发明实施例的再一方面,提供了一种计算机可读介质,其上存储有计算机程序,所述程序被处理器执行时实现上述任一所述的数据处理方法。

[0030] 根据本发明所述提供的方案,上述发明中的一个实施例具有如下优点或有益效果:根据TF-Serving的资源使用情况和JSF请求数据的资源需求量进行负载均衡,可以更合理的使用TF-Serving的GPU资源,使得每台TF-Serving使用的GPU资源基本一致且达到最大化,以此提升整体服务性能和服务稳定性。

[0031] 上述的非惯用的可选方式所具有的进一步效果将在下文中结合具体实施方式加以说明。

附图说明

[0032] 附图用于更好地理解本发明,不构成对本发明的不当限定。其中:

[0033] 图1是根据本发明实施例的一种数据处理系统的主要模块示意图;

[0034] 图2是根据本发明实施例的一种数据处理方法的主要流程示意图;

[0035] 图3是根据本发明实施例的一具体地数据处理方法的流程示意图;

[0036] 图4是本发明实施例可以应用于其中的示例性系统架构图;

[0037] 图5是适于用来实现本发明实施例的移动设备或服务器的计算机系统的结构示意图。

具体实施方式

[0038] 以下结合附图对本发明的示范性实施例做出说明,其中包括本发明实施例的各种细节以助于理解,应当将它们认为仅仅是示范性的。因此,本领域普通技术人员应当认识到,可以对这里描述的实施例做出各种改变和修改,而不会背离本发明的范围和精神。同样,为了清楚和简明,以下的描述中省略了对公知功能和结构的描述。

[0039] 目前部署模型较为主流的方式:基于K8S+容器方式进行部署,并将TF-Serving服务器部署在K8S平台。模型主要运行在TF-Serving中,TF-Serving对外提供grpc服务和http服务,在K8S平台中,深度学习模型在线部署的这两种方式具有不同的缺点,如下所示:

[0040] 1、方式一,受语言及开发人员水平的限制,导致单台服务器处理模型的能力不足,服务可用度较低。另外,在使用GPU批处理时较为复杂,易降低服务的可靠性。单条数据请求服务,无法充分利用GPU算力的优势,在高并发环境下服务在线部署费用昂贵。

[0041] 2、方式二,特别是与JSF共同部署模型到K8S平台中的同一个pod时,易出现TF-Serving死亡导致pod无法重启,无法发挥K8S集群的优势。另外,TF-Serving在使用过程中未提供负载均衡解决方案,因而不能合理使用GPU资源,也无法集群化部署服务。

[0042] 对于本方案涉及的词语,做解释如下:

[0043] TF Serving:TensorFlow Serving是GOOGLE开源的一个服务系统,适用于部署机器学习模型,灵活、性能高、可用于生产环境。

[0044] GPU(Graphics Processing Unit,中文为图形处理器):最初是用在个人电脑、工作站、游戏机和一些移动设备(如平板电脑、智能手机等)上运行绘图运算工作的微处理器。

[0045] K8S:一种可自动实施Linux容器操作的开源平台,能够帮助用户省去应用容器化过程的许多手动部署和扩展操作。即可以将运行Linux容器的多组主机聚集在一起,由Kubernetes帮助轻松高效地管理这些集群。

[0046] GPU在使用时具有以下特点:

[0047] 1、提供多核并行计算的基础结构,且核心数非常多,可以支撑大量数据的并行计算。并行计算或称平行计算是相对于串行计算而言的,一次可执行多个指令,用以提高计算速度及通过扩大问题求解规模,以此解决大型而复杂的计算问题。

[0048] 2、拥有更高的访存速度。

[0049] 3、拥有较高的浮点运算能力,浮点运算能力是关系到处理器的多媒体、3D图形处理的一个重要指标。现有计算机技术中,由于涉及大量多媒体技术的应用,大大增加了浮点数的计算,比如3D图形的渲染等工作,因此浮点运算的能力是考察处理器计算能力的重要

指标。

[0050] 正是因为上述三个特点,GPU非常适合深度学习。神经网络的训练所需的性能、特性,实际上和图形算法非常类似,通常涉及大量参数、激活值、梯度值的缓冲区,其中每个值在每次训练迭代中都会被更新。这些缓冲过大,会超出传统的桌面计算机cache,所以内存带宽通常会成为主要瓶颈。而GPU相比CPU的一个显著优势是其极高的内存带宽,神经网络训练通常不涉及大量的分支运算与复杂的控制指令,所以更适合在GPU上训练。由于神经网络能够被分成多个单独的“神经元”,并且独立于同一层内的其他单元,所以神经网络可以从GPU的并行特性上受益匪浅。

[0051] 参见图1,示出了一种请求处理系统,同样基于K8S+容器方式进行部署,并将TF-Serving服务器部署在K8S平台中,包括客户端服务模块、负载均衡服务模块、服务器监控模块、服务器使用率模块和服务器部署模块,这5个模块相互配合完成整体模型部署方案;

[0052] 1、客户端服务模块,用于将数据处理功能注册到服务中心,以提供微服务接口供JSF客户端调用;接收JSF客户端的数据处理请求,将JSF请求数据放入负载均衡服务模块的内存队列中;以及将负载均衡服务模块传输的处理结果返回给JSF客户端。

[0053] GPU由于其并行计算的特点,处理一条数据与处理一批数据所用的时间基本相同,因而微服务接口的参数优选设置为集合参数,如数组,以批量接收客户端传递的JSF请求数据。另外,客户端服务模块主要的作用是数据引流,因而在部署过程中只需保证其高可用所需最小数量即可。

[0054] 2、服务器监控模块,负责获取每台TF-Serving服务器的CPU、GPU、内存等相关资源使用情况。该服务采用开源的普罗米修斯工具实现,可以从普罗米修斯提供的相关接口获取统计数据。

[0055] 另外,在后续将JSF请求数据发给服务器处理后,若该请求数据所需的资源量,实际大于服务器的可用资源量,则需将该请求数据丢掉,以免影响该服务器处理其他JSF请求数据的处理进度。但若该服务器在一定时长内(如1小时)丢包的数量大于或等于预设数值,如60>50,则触发报警提醒,以对该服务器添加资源量、和/或对JSF请求数据进行限流操作。

[0056] 3、服务器使用率模块,负责从服务器监控模块获取每台服务器的资源使用情况,并使用这些数据计算出每台服务器的资源使用率。考虑模型主要使用到GPU,因而GPU的权重是主要指标,相对于GPU、内存较大。通过大量的测试,优选使用如下公式计算每台服务器的资源使用率(实际权重值可调):

[0057] CPU使用率*0.2+内存使用率*0.1+GPU使用率*0.7=服务器资源使用率

[0058] 在计算到每台服务器的资源使用率后,对service内存中保存的服务器列表,按照从小到大的顺序进行排序,得到服务器序列。

[0059] 4、负载均衡服务模块,用于从内存队列中取出与服务器部署数量等量的JSF请求数据集,确定JSF请求数据集中每个JSF请求数据所需的资源量,按照资源量从大到小的顺序进行排序,得到JSF请求数据序列;获取每台服务器当前的资源使用率,按照资源使用率从小到大的顺序进行排序,得到服务器序列;对JSF请求数据序列和服务器序列进行匹配,以将需求资源量最大的JSF请求数据分配给当前资源使用率最小的服务器,以此实现合理调用TF-Serving服务器的目的。

[0060] 由于客户端服务模块本身也具备负载均衡功能,因而可以将客户端服务模块和

TF-Serving部署在同一个docker容器内,后续两者可以使用127.0.0.1或localhost进行通信,同时无需关注TF-Serving部署时的IP地址,以此简化部署方案。但这种部署也存有缺点,若TF-Serving死亡会导致docker容器无法重启,相应客户端服务模块也无法正常使用,进而影响整体程序的可靠性。

[0061] 作为上一实施方式的优化方案,同时保留上一实施方式的优势,将客户端服务模块和TF-Serving部署到同一pod中的不同docker容器内。但在高并发情况下,这种部署方式会导致JSF客户端资源使用过多,无法合理调度TF-Serving以合理使用GPU资源。

[0062] 以上两种部署方案中,客户端服务模块只是获取JSF请求数据并转发给TF-Serving,因而只需保证客户端服务模块可接收最大并发量的数据即可,以此节省客户端服务模块使用的服务器资源,降低部署成本。为解决上述两种部署方式的缺点,本方案采用了新的负载均衡方式以合理调度TF-Serving,达到合理使用GPU资源、提高TF-Serving高可用性的目的。

[0063] 负载均衡主要有轮询方式、按照服务器资源进行分配请求方式两种,为更合理的使用TF-Serving的GPU资源,优选采用第二种方式。且调用过程无需关注每台TF-Serving的IP地址,而是直接将JSF请求数据发送到负载均衡服务模块,再由负载均衡服务模块转发到TF-Serving服务器。前述两种方式相对负载均衡方式:1)部署在同一个docker容器,服务器资源虽使用较为均衡,但是GPU使用率一致无法达到最大化,导致并发量比较低,且服务的稳定性也不如本方案。2)部署到同一pod中的不同docker容器,相对1)有所提高,但并发量及GPU合理使用情况都不如本方案。

[0064] 5、服务器部署模块,用于处理负载均衡服务模块传输的JSF请求数据,之后将处理结果通过负载均衡服务模块发给客户端服务模块。TF-Serving与JSF客户端之间采用grpc方式通讯,具体地TCP长连接方式,相比HTTP通讯方式,TCP长连接可以节省建立连接时间以及更高的数据传输效率。

[0065] 参见图2,示出的是本发明实施例提供的一种数据处理方法的主要流程图,包括如下步骤:

[0066] S201:接收客户端的数据处理请求,将请求数据放入负载均衡的内存队列中,从内存队列中取出与服务器部署数量等量的请求数据集;

[0067] S202:确定所述请求数据集中每个请求数据所需的资源量,按照资源量从大到小的顺序进行排序,得到请求数据序列;

[0068] S203:计算每台服务器当前的资源使用率,按照资源使用率从小到大的顺序进行排序,得到服务器序列;

[0069] S204:对请求数据序列和服务器序列进行匹配,以将需求资源量最大的请求数据分配给当前资源使用率最小的服务器进行处理,之后将接收自服务器的处理结果返回给客户端。

[0070] 上述实施方式中,对于步骤S201～S202,同一时刻调用客户端服务模块的客户端数量可能不止一个,这些客户端传输的JSF请求数据会同时到达客户端服务模块,首先将这些JSF请求数据放入负载均衡服务模块的内存队列中,按照TF-Serving的部署数量,从内存队列中获取出等量的JSF请求数据集。进一步的,还可以按照数据的长度、接收时间,对不同JSF请求数据进行分类。

[0071] 由于每个JSF请求数据的数据量和字节数不一致,因而所需使用的资源量也不相同,优选针对数据所需使用的资源量,按照从大到小的顺序进行排序,生成JSF请求数据序列。

[0072] 对于步骤S203,同样,获取每台服务器的CPU、GPU、内存等相关资源使用情况。该服务采用开源的普罗米修斯工具实现,可以从普罗米修斯提供的相关接口获取统计数据。考虑模型主要使用到GPU,因而GPU的权重是主要指标,相对于GPU、内存较大。通过大量的测试,优选使用如下公式计算每台服务器的资源使用率:

[0073] CPU使用率*0.2+内存使用率*0.1+GPU使用率*0.7＝服务器资源使用率

[0074] 在计算到每台服务器的资源使用率后,对service内存中保存的服务器列表,按照从小到大的顺序进行排序,得到服务器序列。

[0075] 对于步骤S204,对排序后的JSF请求数据序列与服务器排序进行匹配,使用需求资源量最大的JSF请求数据与TF-Serving资源使用率最低的进行匹配,以此类推。对应后将JSF请求数据发送到关联的TF-Serving,再将TF-Serving处理后的结果通过负载均衡发送到对应的请求客户端,整体参见图3所示。

[0076] 另外,在后续将JSF请求数据发给服务器处理后,若该请求数据所需的资源量,实际大于服务器的可用资源量,则需将该请求数据丢掉,以免影响后续JSF请求数据的处理进度。但若该服务器在一定时长内(如1小时)丢包的数量大于或等于预设数值,如60>50,则触发报警提醒,以对该服务器添加资源量、和/或对请求数据进行限流操作。

[0077] 根据上述实施例所提供的方法,根据TF-Serving的资源使用情况和JSF请求数据的资源需求量进行负载均衡,可以更合理的使用TF-Serving的GPU资源,使得每台TF-Serving使用的GPU资源基本一致且达到最大化,以此提升整体服务性能和服务稳定性。

[0078] 图4示出了可以应用本发明实施例的示例性系统架构400,包括终端设备401、402、403,网络404和服务器405(仅仅是示例)。

[0079] 终端设备401、402、403可以是具有显示屏并且支持网页浏览的各种电子设备,安装有各种通讯客户端应用,用户可以使用终端设备401、402、403通过网络404与服务器405交互,以接收或发送消息等。

[0080] 网络404用以在终端设备401、402、403和服务器405之间提供通信链路的介质。网络404可以包括各种连接类型,例如有线、无线通信链路或者光纤电缆等等。

[0081] 服务器405可以是提供各种服务的服务器,用于执行接收客户端调用,生成请求数据序列和服务器序列,并对两者进行匹配操作。需要说明的是,本发明实施例所提供的方法一般由服务器405执行,相应地,系统一般设置于服务器405中。

[0082] 应该理解,图4中的终端设备、网络和服务器的数目仅仅是示意性的。根据实现需要,可以具有任意数目的终端设备、网络和服务器。

[0083] 下面参考图5,其示出了适于用来实现本发明实施例的终端设备的计算机系统500的结构示意图。图5示出的终端设备仅仅是一个示例,不应对本发明实施例的功能和使用范围带来任何限制。

[0084] 如图5所示,计算机系统500包括中央处理单元(CPU)501,其可以根据存储在只读存储器(ROM)502中的程序或者从存储部分508加载到随机访问存储器(RAM)503中的程序而执行各种适当的动作和处理。在RAM 503中,还存储有系统500操作所需的各种程序和数据。

CPU 501、ROM 502以及RAM 503通过总线504彼此相连。输入/输出(I/O)接口505也连接至总线504。

[0085] 以下部件连接至I/O接口505:包括键盘、鼠标等的输入部分506;包括诸如阴极射线管(CRT)、液晶显示器(LCD)等以及扬声器等的输出部分507;包括硬盘等的存储部分508;以及包括诸如LAN卡、调制解调器等的网络接口卡的通信部分509。通信部分509经由诸如因特网的网络执行通信处理。驱动器510也根据需要连接至I/O接口505。可拆卸介质511,诸如磁盘、光盘、磁光盘、半导体存储器等等,根据需要安装在驱动器510上,以便于从其上读出的计算机程序根据需要被安装入存储部分508。

[0086] 特别地,根据本发明公开的实施例,上文参考流程图描述的过程可以被实现为计算机软件程序。例如,本发明公开的实施例包括一种计算机程序产品,其包括承载在计算机可读介质上的计算机程序,该计算机程序包含用于执行流程图所示的方法的程序代码。在这样的实施例中,该计算机程序可以通过通信部分509从网络上被下载和安装,和/或从可拆卸介质511被安装。在该计算机程序被中央处理单元(CPU)501执行时,执行本发明的系统中限定的上述功能。

[0087] 需要说明的是,本发明所示的计算机可读介质可以是计算机可读信号介质或者计算机可读存储介质或者是上述两者的任意组合。计算机可读存储介质例如可以是——但不限于——电、磁、光、电磁、红外线、或半导体的系统、装置或器件,或者任意以上的组合。计算机可读存储介质的更具体的例子可以包括但不限于:具有一个或多个导线的电连接、便携式计算机磁盘、硬盘、随机访问存储器(RAM)、只读存储器(ROM)、可擦式可编程只读存储器(EPROM或闪存)、光纤、便携式紧凑磁盘只读存储器(CD-ROM)、光存储器件、磁存储器件、或者上述的任意合适的组合。在本发明中,计算机可读存储介质可以是任何包含或存储程序的有形介质,该程序可以被指令执行系统、装置或者器件使用或者与其结合使用。而在本发明中,计算机可读的信号介质可以包括在基带中或者作为载波一部分传播的数据信号,其中承载了计算机可读的程序代码。这种传播的数据信号可以采用多种形式,包括但不限于电磁信号、光信号或上述的任意合适的组合。计算机可读的信号介质还可以是计算机可读存储介质以外的任何计算机可读介质,该计算机可读介质可以发送、传播或者传输用于由指令执行系统、装置或者器件使用或者与其结合使用的程序。计算机可读介质上包含的程序代码可以用任何适当的介质传输,包括但不限于:无线、电线、光缆、RF等等,或者上述的任意合适的组合。

[0088] 附图中的流程图和框图,图示了按照本发明各种实施例的系统、方法和计算机程序产品的可能实现的体系架构、功能和操作。在这点上,流程图或框图中的每个方框可以代表一个模块、程序段、或代码的一部分,上述模块、程序段、或代码的一部分包含一个或多个用于实现规定的逻辑功能的可执行指令。也应当注意,在有些作为替换的实现中,方框中所标注的功能也可以以不同于附图中所标注的顺序发生。例如,两个接连地表示的方框实际上可以基本并行地执行,它们有时也可以按相反的顺序执行,这依所涉及的功能而定。也要注意的是,框图或流程图中的每个方框、以及框图或流程图中的方框的组合,可以用执行规定的功能或操作的专用的基于硬件的系统来实现,或者可以用专用硬件与计算机指令的组合来实现。

[0089] 描述于本发明实施例中所涉及到的模块可以通过软件的方式实现,也可以通过硬

件的方式来实现。所描述的模块也可以设置在处理器中,例如,可以描述为:一种处理器包括客户端服务模块、负载均衡服务模块、服务器部署模块。其中,这些模块的名称在某种情况下并不构成对该模块本身的限定,例如,负载均衡服务模块还可以被描述为“负载均衡模块”。

[0090] 作为另一方面,本发明还提供了一种计算机可读介质,该计算机可读介质可以是上述实施例中描述的设备中所包含的;也可以是单独存在,而未装配入该设备中。上述计算机可读介质承载有一个或者多个程序,当上述一个或者多个程序被一个该设备执行时,使得该设备包括:

[0091] 提供微服务接口供客户端调用,接收客户端的数据处理请求,将请求数据放入负载均衡服务模块的内存队列中;以及将所述负载均衡服务模块传输的处理结果返回给客户端;

[0092] 从内存队列中取出与服务器部署数量等量的请求数据集,确定所述请求数据集中每个请求数据所需的资源量,按照资源量从大到小的顺序进行排序,得到请求数据序列;计算每台服务器当前的资源使用率,按照资源使用率从小到大的顺序进行排序,得到服务器序列;对请求数据序列和服务器序列进行匹配,以将需求资源量最大的请求数据分配给当前资源使用率最小的服务器;

[0093] 处理所述负载均衡服务模块传输的请求数据,之后将处理结果通过所述负载均衡服务模块发给所述客户端服务模块。

[0094] 根据本发明实施例的技术方案,根据TF-Serving的资源使用情况和JSF请求数据的资源需求量进行负载均衡,可以更合理的使用TF-Serving的GPU资源,使得每台TF-Serving使用的GPU资源基本一致且达到最大化,以此提升整体服务性能和服务稳定性。

[0095] 上述具体实施方式,并不构成对本发明保护范围的限制。本领域技术人员应该明白的是,取决于设计要求和其他因素,可以发生各种各样的修改、组合、子组合和替代。任何在本发明的精神和原则之内所作的修改、等同替换和改进等,均应包含在本发明保护范围之内。

图1

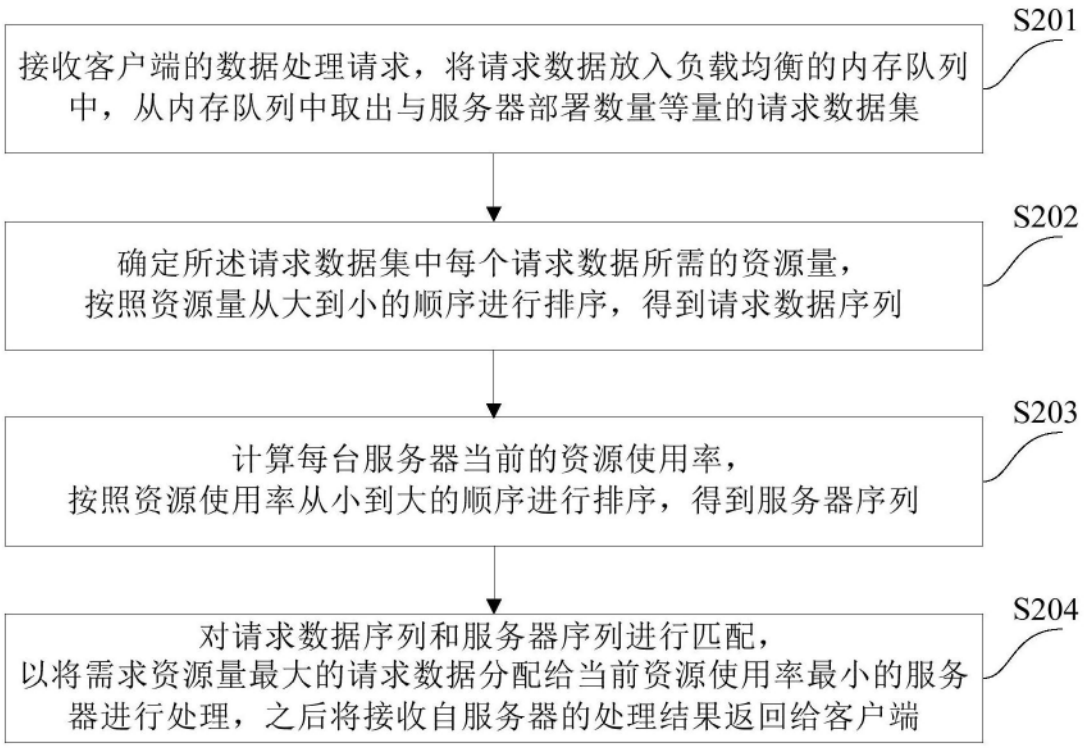

S201
接收客户端的数据处理请求，将请求数据放入负载均衡的内存队列中，从内存队列中取出与服务器部署数量等量的请求数据集
S202
确定所述请求数据集中每个请求数据所需的资源量，按照资源量从大到小的顺序进行排序，得到请求数据序列
S203
计算每台服务器当前的资源使用率，按照资源使用率从小到大的顺序进行排序，得到服务器序列
S204
对请求数据序列和服务器序列进行匹配，以将需求资源量最大的请求数据分配给当前资源使用率最小的服务器进行处理，之后将接收自服务器的处理结果返回给客户端

图2

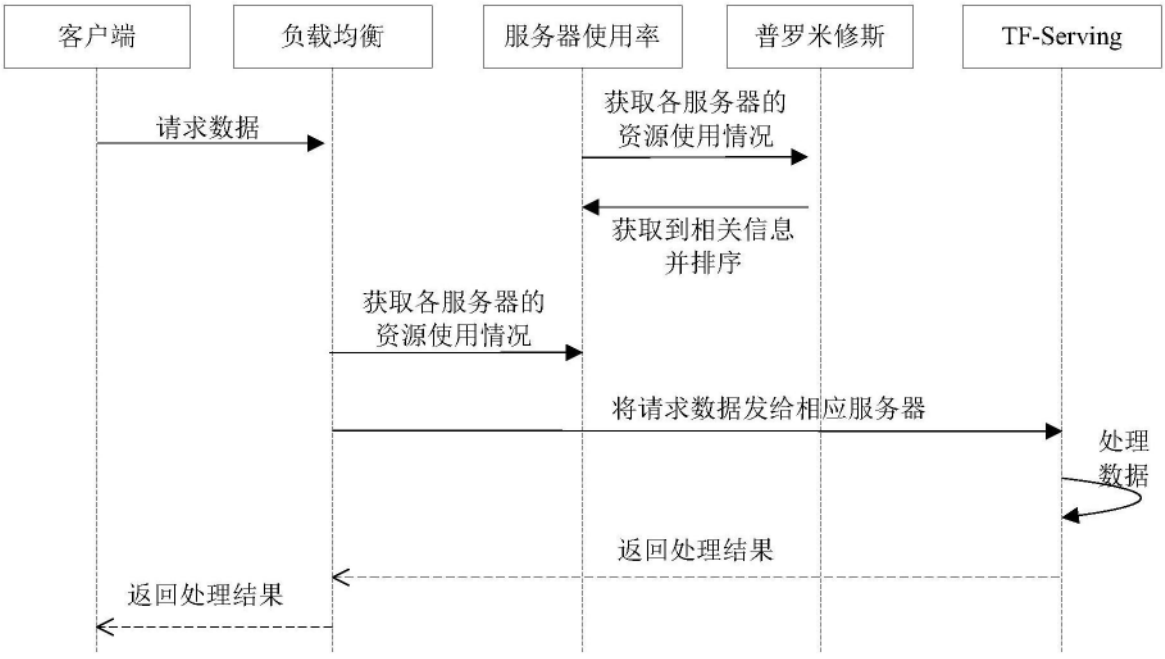

客户端
负载均衡
服务器使用率
普罗米修斯
TF-Serving
请求数据
获取各服务器的资源使用情况
获取到相关信息并排序
获取各服务器的资源使用情况
将请求数据发给相应服务器
处理数据
返回处理结果
返回处理结果

图3

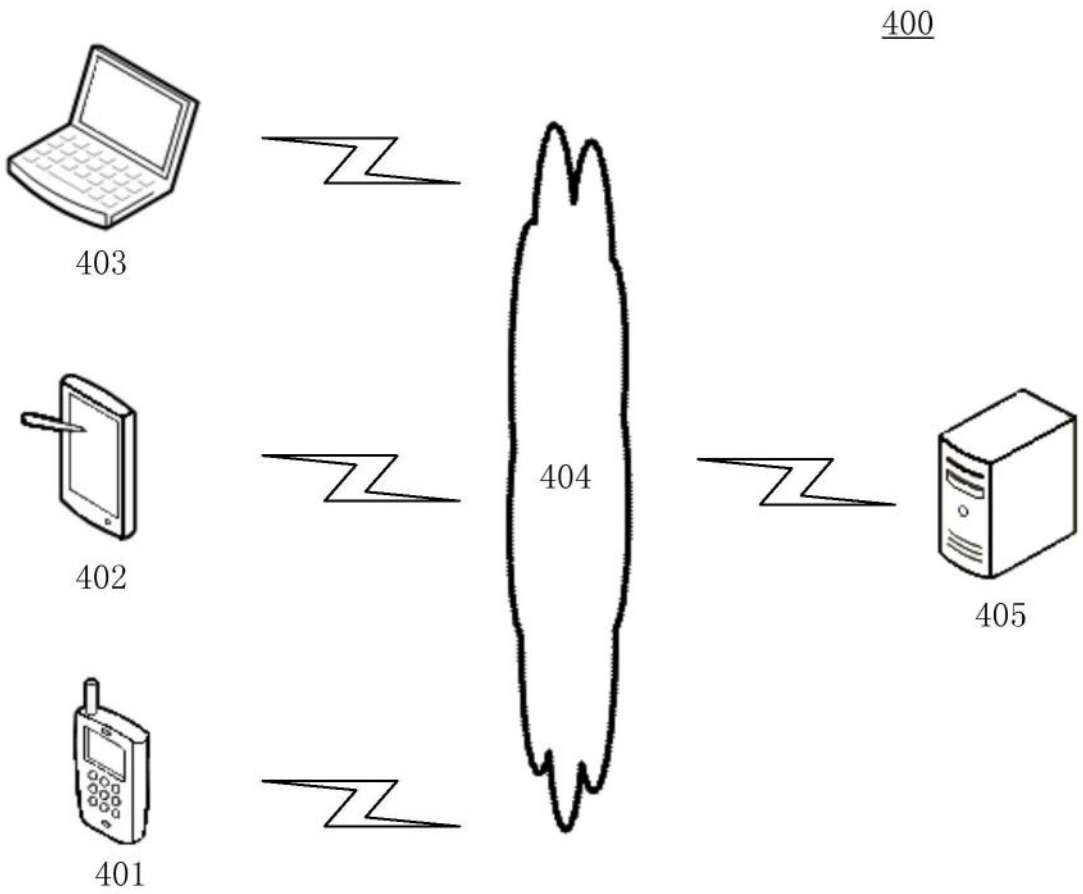


图4

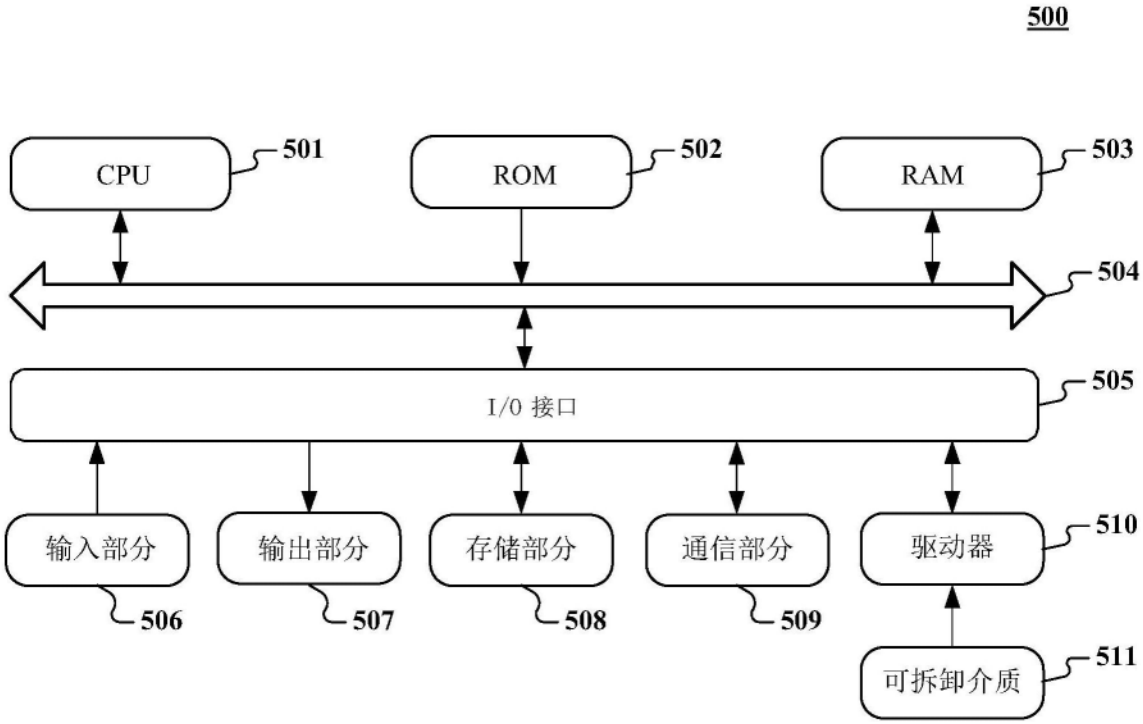

500
CPU
501
ROM
502
RAM
503
504
I/0 接口
505
输入部分
506
输出部分
507
存储部分
508
通信部分
509
驱动器
510
可拆卸介质
511

图5